Parallel Processing in a Computationally-Intensive Workload: Update 05/15/92

Robert J. Bergeron
Computer Sciences Corporation
NASA Ames Research Center
Moffett Field, CA 94035, USA

Abstract

This paper updates NAS Cray Y-MP performance results for computationally intensive workloads employing autotasking. The results indicate the post memory-upgrade UNICOS operating system continued to efficiently execute workloads containing autotasked programs. For workloads oversubscribing memory and performing a fixed number of floating point operations, autotasking a portion of the workload reduced the elapsed time relative to singletasked execution of the same workload. Factors influencing the size of the elapsed time reduction include the amount of workload idle, the number of jobs in the workload and the size of jobs in the workload.

1.0 Introduction

This paper provides an update of the autotasking performance of UNICOS 6.0 on heterogeneous workloads characteristic of the Numerical Aerodynamic Simulation (NAS) Y-MP. For the NAS workload, the Y-MP memory upgrade from 128 MW to 256 MW allows execution of at least one large (> 60 MW) user job at all times and has required an increase in the number of jobs allowed to be memory-resident and executable. The increased memory motivated this update since it has produced a significant impact on the job mix and administration of the NAS workload.

NAS administrators employ a scheduling algorithm to deliver the Network Queueing System (NQS) batch jobs to the bottom of the UNICOS run queue. This algorithm ensures sufficient space for each batch job by assuming that the job requires the maximum memory available to the queue. Experience with the algorithm indicated that limiting the maximum executable jobs to 12 (the limit for the 128 MW Y-MP) frequently undersubscribed the 256 MW memory. Such undersubscription occurs when users would submit a 50 MW job to the 64 MW queue because it does not fit into the 32 MW queue; the scheduling algorithm had to assume that the job required 64 MW since the job was in the 64 MW queue. The administrative remedy has been to increase the batch job limit to 20; in practice, about 15 batch jobs are in execution during the off-prime hours. System joblogs indicate that the generation of significant idle during off-prime hours is a transient condition, usually brought about by the execution of several big jobs.

A related effect of the memory upgrade, increased demand for SRFS space, motivated an additional NAS modification, the reduction of the swap cache located on SSD from 32 MW to 10 MW. During periods of swapping, this change could allow more idle CPU time for use by autotasking since transfer of the executable images from swap into memory would tend to occur at disk speeds.

Autotasking a portion of a supercomputer's workload can be effective in increasing throughput for workloads displaying a non-negligible fraction of idle time. For workloads containing a small amount of idle time, the autotasking overhead and less efficient usage of the CPUs will prevent throughput increases. Evaluation of autotasking should thus compare the performance of a singletasked workload (containing idle time) with that of the same workload with an autotasked component.

The singletasked workload can be arranged to display a fixed number of floating point operations or to display a fixed idle component. Workloads with a fixed number of operations can display a varying amount of idle depending upon the scheduler options. NAS employs a different set of scheduler options for prime and off-prime service. To claim a performance advantage, the autotasked version of this type of workload should complete before its singletasked counterpart. Workloads with a fixed idle component require careful scheduling to ensure the proper amount of idle during execution. To claim a performance advantage, the autotasked version of this type of workload should perform more work than its singletasked counterpart during the period designated as high idle. The autotasked workload will perform more work by reducing the high idle period to a low or zero idle period.

A previous report (Bergeron, 1990) described the performance of a workload which executed first with all codes singletasked and then with some of the codes autotasked and the remainder of the codes singletasked. The distinguishing feature of this workload was that for high-overhead implementations of autotasking, the autotasked workload always required more wall clock time than the singletasked workload. For low-overhead implementations of autotasking, the autotasked workload always required less wall clock time than the singletasked workload. The "always" in this context meant under a variety of UNICOS scheduling options. The mix of autotasked jobs did not reflect an optimal throughput. Rather, the mix tested the ability of the operating system to maintain the extra CPU and system time associated with autotasking at a level low enough to allow the autotasked workload to complete in less wall clock time than the singletasked workload. The value of the workload was that it provided a "go-nogo" verdict on the current implementation of autotasking.

Since the previous report, additional tests of the 128 MW machine were executed on 07/30/91 and 11/13/91. These results, along with the last result from the previous report provide the results shown in the table.

Table 1. 169 MW Model Workload Pre-Upgrade Results

Date	03/13/91		07/30/91		11/13/91	
Workload	Single	Auto	Single	Auto	Single	Auto
Elapsed Time	1219	1173	1236	1210	1190	1086
CPU Time	7319	7891	7009	7849	6961	7643
System Time	96	177	48	151	43	128
Idle Time	2337	1316	2830	1680	2515	912

System GFLOPS	0.903	0.939	0.891	0.910	0.925	1.014

For each experiment, the elapsed time for the autotasked workload was less than the corresponding elapsed time for the singletasked workload. System performance, as measured by the wall clock time required to execute a fixed number of floating point operations, indicates that UNICOS maintained the autotasking performance improvement discussed in the previous report. Table 1 also shows increased CPU time and system time for the autotasked workload relative to the corresponding singletasked quantities.

Autotasking increases CPU time because the autotasked processors:

- wait on semaphores during synchronization,
- execute extra code associated with ordering the processors,
- process shorter vectors as multiple processors do vector work, and
- suffer memory conflicts as multiple CPUs reference the same
- memory area.

Autotasking increases system time because the autotasked processors will invoke a system call *resch* to surrender themselves to the operating system when the master processor has no work for them.

To test the autotasking performance of the larger memory workload, synthetic workloads consisting of NAS Y-MP production codes have been run on a dedicated machine. The jobs in this workload were the only jobs in the system. The experiments employed the same machine configuration used in NAS Y-MP production periods. Oversubscription of the 256 MW memory produced time intervals dominated by the execution of several large-memory jobs. The amount of memory remaining was insufficient to allow all of the other CPUs to execute and some of the CPUs idled. The autotasked programs were able to use these idle CPUs effectively to increase workload throughput.

The memory upgrade to 256 MW required larger memory programs to generate the idle for the autotasked programs. The following sections describe the revisions to the earlier model of the NAS Y-MP workload and the results of the revised workload executions on the 6.1.5 version of the UNICOS operating system.

2.0 Modelling NAS Y-MP Workload

The workload performance of supercomputers executing a set of mixed (single CPU and multiple CPU) programs depends upon workload idle time. NAS administrators try to minimize Y-MP CPU idle time by carefully allocating system resources; however, memory oversubscription will occasionally lead to idle CPUs. This paper reports two cases for the 256 MW Y-MP: a 12-job workload requiring 287 MW and a 15-job workload requiring 303 MW.

Since the first case is an extension of the previous 12-job 169 MW workload developed for the 128 MW Y-MP, Table 2 lists the characteristics of the 169 MW workload for reference.

Table 2. 169 MW Model Workload Parameters

Code	CPU (Sec)	Size (MW)	Rate (MFLOPS)	FLOPS %	Comments
C01	595	56	222	11.6	Autotasked
C02	625	52	164	9.0	Singletasked
C03	602	9	161	8.5	Singletasked
C04	601	7	187	9.9	Autotasked
C05	601	7	187	9.9	Autotasked
C06	601	7	187	9.9	Autotasked
C07	617	5	165	9.0	Singletasked
C08	603	7	133	7.1	Singletasked
C09	143	0.9	122	6.1	Singletasked, 4 copies serially
C10	122	1.7	125	6.7	Singletasked, 5 copies serially
C11	82	0.8	131	6.6	Autotasked, 7 copies serially
C12	605	1.6	105	5.7	Singletasked, SSD sync I/O
Totals	7179	169		100	

This workload performed 1135 billion floating point operations, and since the UNICOS kernel requires about 6 MW, this workload oversubscribed the 128 Y-MP memory by about 47 MW.

Table 3 shows dedicated 8-CPU speedups and efficiencies demonstrated by the codes designated as autotasked in Table 2.

Table 3. Performance of Autotasked Codes in Dedicated Time

UNICOS	6.0	
CODE	Speedup	Efficiency
C01	7.67	.959
C04	3.49	.436
C05	3.49	.436
C06	3.49	.436
C11	3.12	.390

The NAS workload does not normally execute 5 autotasked jobs simultaneously; this large number of autotasked jobs provides a severe challenge to the ability of autotasking to deliver a throughput performance increase, since the autotasked jobs must display limited overhead.

Table 4 shows the workload for the 256 MW Y-MP. This workload was constructed by doubling the size of several jobs to obtain a memory oversubscription in rough agreement with that of the 169 MW workload.

Table 4. 287 MW Model Workload Parameters

Code	CPU (Sec)	Size (MW)	Rate (MFLOPS)	FLOPS %	Comments
C01	590	84	222	11.6	Autotasked
C02	606	105	164	8.8	Singletasked
C03	596	22	160	8.4	Singletasked
C04	602	15	187	10.0	Autotasked
C05	602	15	187	10.0	Autotasked
C06	602	15	187	10.0	Autotasked
C07	617	5	165	9.0	Singletasked
C08	603	7	133	7.1	Singletasked
C09	143	0.9	122	6.2	Singletasked, 4 copies serially
C10	122	1.7	125	6.7	Singletasked, 5 copies serially
C11	82	0.8	131	6.6	Autotasked, 7 copies serially

C12	605	1.6	105	5.6	Singletasked, SSD sync I/O
Total	7179	287.3		100	

Codes C01-C06 differ as follows from the Table 2 counterparts:

- C01 increased memory requirements from 56 MW to 84 MW;
- C02 increased memory requirements from 52 MW to 105 MW;
- C03 (10MW) replaced by a similar code requiring 22 MW; and
- C04, C05, and C06 increased memory from 8 MW to 15 MW.

The maximum timesteps for new versions of codes C01-C06 were chosen to require 600 CPU seconds for execution. This workload performed about 1130 billion floating point operations and oversubscribed the 256 MW memory by about 37 MW, approximately equal to the 47 MW oversubscribed by the 169 MW workload. Execution of this workload employed the UNICOS scheduler parameters developed by NAS for the 12-job 128 MW workloads during the off-prime hours. Off-Prime parameters disable the UNICOS hog constraints so that once a process has obtained sufficient memory and a connection to a CPU, only a kernel or I/O interrupt can disconnect the process. These parameters promote the execution of computationally-intensive jobs.

The 15-job workload arose in response to the the NAS administrative workload modifications for the 256 MW machine as discussed in Section 1. Examination of the system joblogs indicated that the number of batch jobs fluctuated around 15 for off-prime operation, and that periods of swapping-induced idle generally occurred when several large-memory jobs were executing. The following workload attempts to model this behavior by increasing the number of executable jobs and by employing a memory distribution representative of the job distribution observed for periods of memory oversubscription and idle. Table 5 shows this workload, consisting of 15 jobs and requiring 303 MW of memory.

Table 5. 303 MW Model Workload Parameters

Code	CPU (Sec)	Size (MW)	Rate (MFLOPS)	FLOPS %	Comments
C01	595	56	222	9.3	Autotasked
C02	625	52	164	7.2	Singletasked
C03	596	22	160	6.7	Singletasked
C04	601	32	187	7.9	Autotasked
C05	601	15	187	7.9	Autotasked
C06	601	15	187	7.9	Autotasked
C07	617	5	165	7.0	Singletasked
C08	603	7	133	5.6	Singletasked
C09	143	0.9	122	4.9	Singletasked, 4 copies serially
C10	122	1.7	125	5.3	Singletasked, 5 copies serially
C11	82	5.6	131	5.3	Autotasked, 7 copies serially
C12	605	1.6	105	4.5	Singletasked, SSD sync I/O
C13	595	56	222	9.3	Autotasked
C14	596	22	160	6.7	Singletasked
C15	605	1.6	105	4.5	Singletasked, SSD sync I/O
Total	8996	302.9		100	

The 303 MW workload has the following features:

- Duplication of the code C01 from the 169 MW workload; C13 designates this duplicate;
- Duplication of the code C03; C14 designates this duplicate;
- Duplication of the code C12; C15 designates this duplicate;
- C02 decreased memory requirements from 105 MW to 52 MW; and
- C04 increased memory from 15 MW to 32 MW.

This workload performed 1425 billion floating point operations and oversubscribed memory by about 53 MW. Execution of this workload employed the batch scheduler parameters developed by NAS for the 15-job 256 MW workloads.

3.0 Results

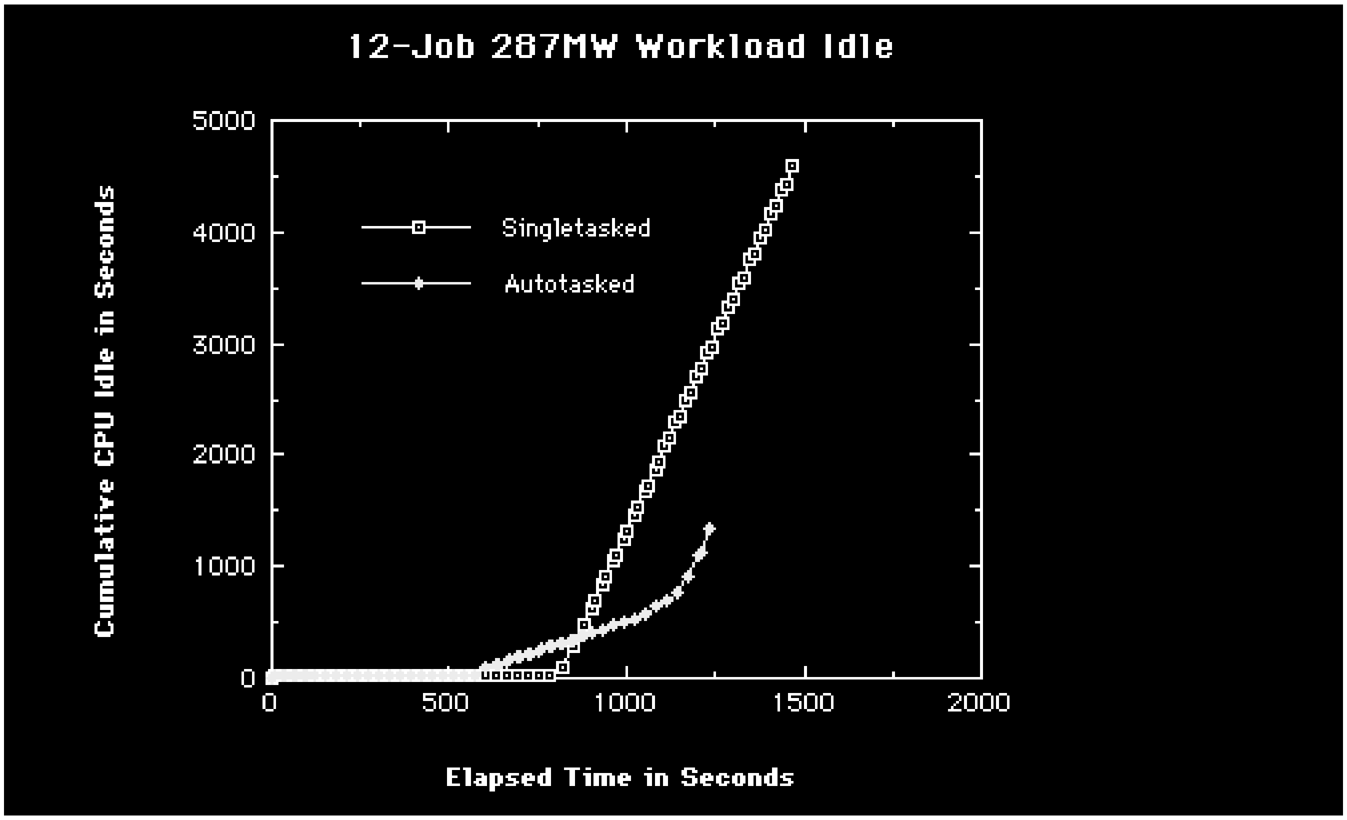
This section presents a discussion of the performance of the 287 MW and 303 MW workloads. These workloads executed in a dedicated environment to provide a controlled investigation of autotasking-generated throughput improvements. Since the workload consisted of a fixed number of jobs, a throughput improvement would correspond to a reduction in elapsed time.

Execution began by invoking a UNIX script which recorded the date, initiated execution of the 12 (or 15) scripts in background, paused until all 12 (or 15) background scripts had completed, and then recorded the completion date. Each of the 12 (or 15) background scripts initiated execution of a workload job and requested accounting information for the job. Accounting logs provide a wealth of job execution data including elapsed time, CPU time, system time and idle (semaphore wait) time. Executables in the singletasked workload corresponded to singletasked versions of the programs described in Section 2.1. The autotasked workload substituted autotasked executables corresponding to the 5 autotasked programs. Execution of all autotasked jobs use the 8-CPU default for autotasked jobs.

A monitoring script executed every 30 seconds to record a history of the CPU time accumulated by the programs in the workload. The script also provided a snapshot of main memory, swapping information, and system counters.

3.1 Performance of the 12-job 287 MW Workload

The following figure shows the growth of cumulative idle for the 12-job 287 MW workloads. These workloads used the batch scheduler parameters developed by NAS for the 12-job 128 MW workloads.



The singletasked workload idle growth remained at zero until 5 jobs completed execution at 850 seconds. Eleven jobs completed execution at 878 seconds and subsequent idle growth reflected 7 idle CPUs. Completion of the 56 MW C01 at 1471 seconds signalled the completion of the singletasked workload.

Idle growth for the autotasked workload remained at zero until 571 seconds. By this time, the 1 MW C11 and 52 MW C01 codes had completed execution. The remaining three autotasked programs wrote plot and restart files as they neared completion and the idle grew from about 571 second to 630 seconds as no autotasked program was available to utilize the free CPUs. Only 7 jobs remained in execution beyond 630 seconds and subsequent idle growth reflected 5 idle CPUs. Completion of the 2 MW C12 signalled the completion of the autotasked workload.

The autotasked workload elapsed time was 1212 seconds, 18% less than that of the singletasked workload. Autotasking makes its contribution to the performance improvement of the workload by using the idle time to generate useful FLOPs. At 878 seconds, shown in the figure as the time when the singletasked workload begins its monotonic increase in idle, the singletasked workload has completed about 91% of workload FLOPS, whereas the autotasked workload has completed about 78% of workload FLOPS. The relative amounts of work completed at this time show that the autotasked workload pays a price for executing the parallel jobs. The figure indicates that the Off-Prime scheduler parameters generated workload idle towards the end of the workload completion and it is in this period that the performance of the autotasked workload overtakes that of the singletasked workload.

A simple estimate of maximum throughput involving the replacement of the 0 MFLOP idle time in both workloads by a CPU performing at the average workload rate results in the following calculation:

Singletasked Workload:

Maximum Throughput (Billions of floating point operations)

=1101+(.909/8)*4639=1101+527=1628

Autotasked Workload:

Maximum Throughput (Billion of floating point operations)

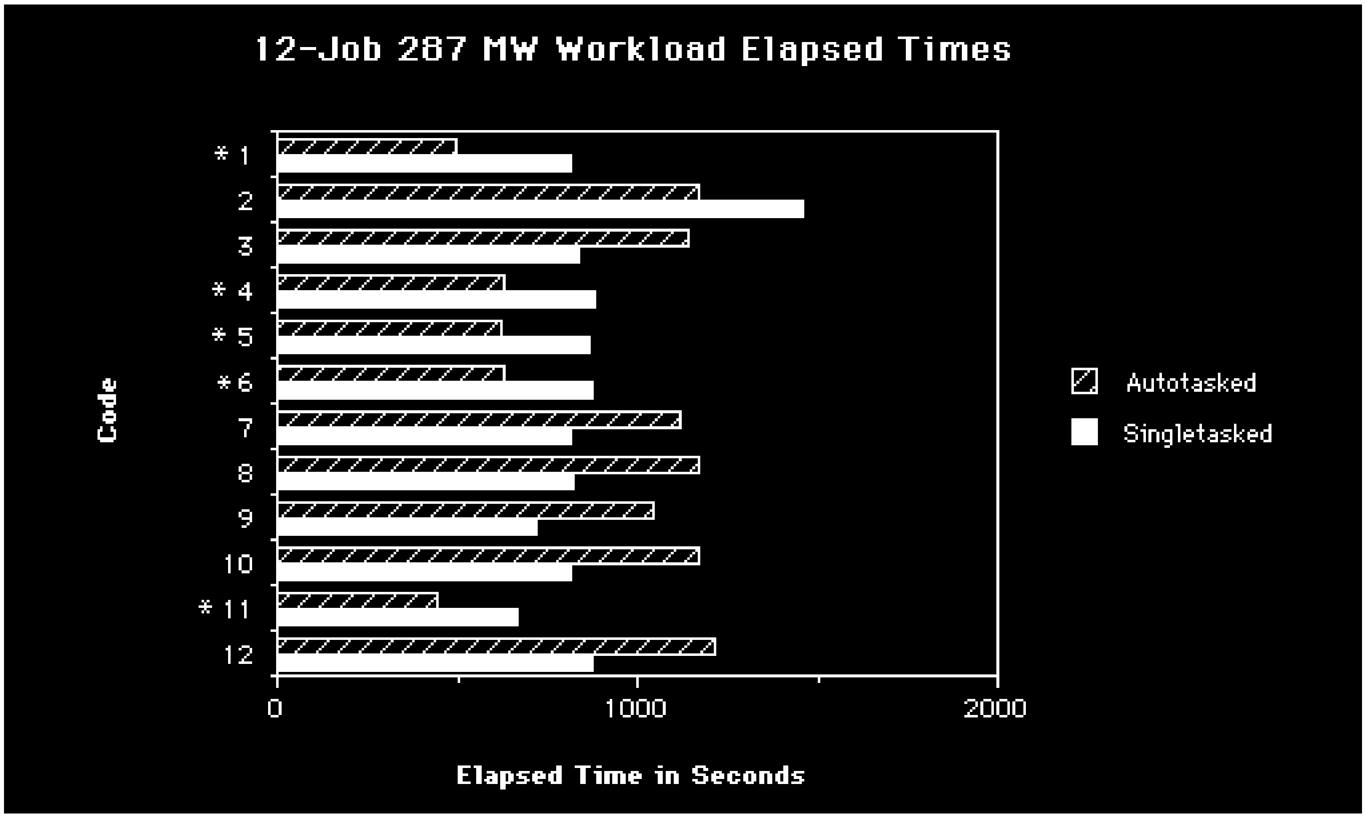
=1101+(.939/8)*3429=1101+402=1503

Idle time for the autotasked workload has two components: the idle time accumulated to completion at 1212 seconds and the additional time from 1212 seconds to 1471 seconds. The calculation shows that the autotasked workload can achieve about 92% of the maximum singletasked throughput.

The above calculation assigns the full singletasked performance rate to the entire singletasked idle time. However, during the idle period for the singletasked workload, at least 52 MW of main memory are occupied. Such memory usage would prevent full utilization of all the singletasked idle time since some of the codes would not fit in memory. For the autotasked workload, 60% of the idle component (from 1212 seconds to 1471 seconds) would involve 100% of the memory available; moreover, autotasked jobs could reduce any idle time due to memory oversubscription during the first 40%.

The autotasked job mix was intended to test the autotasking implementation by providing a highly parallel component: this mix devoted about 41% of the user CPU time (corresponding to 48% of the user FLOPS) to parallel processing. The previous paper (Bergeron, 1990) displayed a workload in which one autotasked job executed continuously until all singletasked jobs completed; the autotasked workload reduced idle growth by a factor of 2 and improved throughput by about 4%.

The following figure displays elapsed times for the 287 MW workload. The figure indicates the elapsed time for each code for execution in the singletasked and autotasked workloads. Asterisks label the autotasked codes.

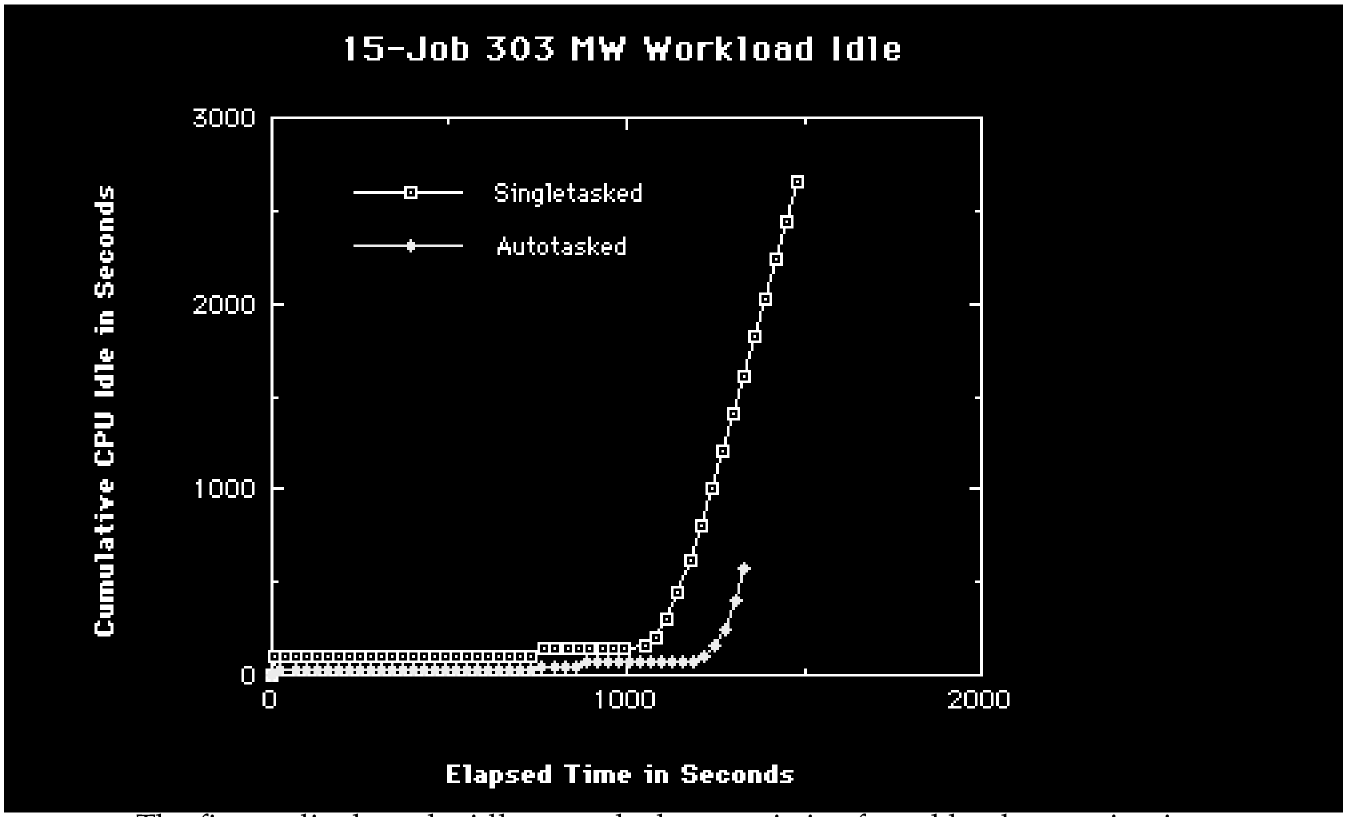


The figure shows that autotasking reduced elapsed time for all autotasked codes. The high efficiency displayed by the autotasked jobs, as shown in Table 2, implied a small number of self-driven I/O interrupts. These jobs tend to finish first in a workload of equal priority jobs because they can obtain additional CPUs for parallel work whenever the singletasked jobs give up the their CPUs. If the autotasked jobs were to monopolize the CPU, their elapsed times would be much less. The figure shows that the autotasked jobs share the CPUs with the singletasked jobs. However, their "finish first" tendency leads to an alteration in the order of job completion relative to the singletasked workload.

The figure shows that autotasking reduced elapsed time for all autotasked codes and increased the elapsed time for all singletasked jobs, except the 52 MW C02. In the singletasked workload, the UNICOS scheduler elected to place C02 on the swap disk until completion of other jobs freed sufficient memory. In the autotasked workload, the UNICOS scheduler also elected to place C02 on the swap disk; however, the smaller amount of wall clock time required by the autotasked jobs freed 52 MW of memory sooner in the autotasked workload and permitted earlier completion of C02.

3.2 Performance of the 15-job 303 MW Workload

The following figure shows the growth of cumulative idle for the 15-job 303 MW workloads. These workloads used the batch scheduler parameters developed by NAS for the 15-job 256 MW workloads.



The figure displays the idle growth characteristic of workload execution in dedicated time with the Off-Prime parameters: a period of near-zero idle followed by a rapid increase in the CPU idle as the completion of jobs reduces the number of executing jobs below the number of available CPUs.

For the singletasked workload, this reduction occurs at 1087 seconds and CPU idle rises until the workload completes at 1481 seconds. The system logs indicated that UNICOS tends to fill available memory with a few big jobs and to pack the smaller jobs around them. For 303 MW workload, the big codes C01, C03, C13, and C14 occupy dominant memory locations and the big C02 must wait until several of these have completed to obtain sufficient memory.

For the autotasked workload, the reduction of executing jobs below the number of available CPUs occurs at 1172 seconds initiates a sharp increase in the cumulative idle. This workload completes at 1326 seconds.

An estimate for maximum throughput similar to that performed for the 287 MW workload yields the following:

Singletasked Workload:

Maximum Throughput (Billions of floating point operations)

=1425+(0.962/8)*2650=1425+319=1744

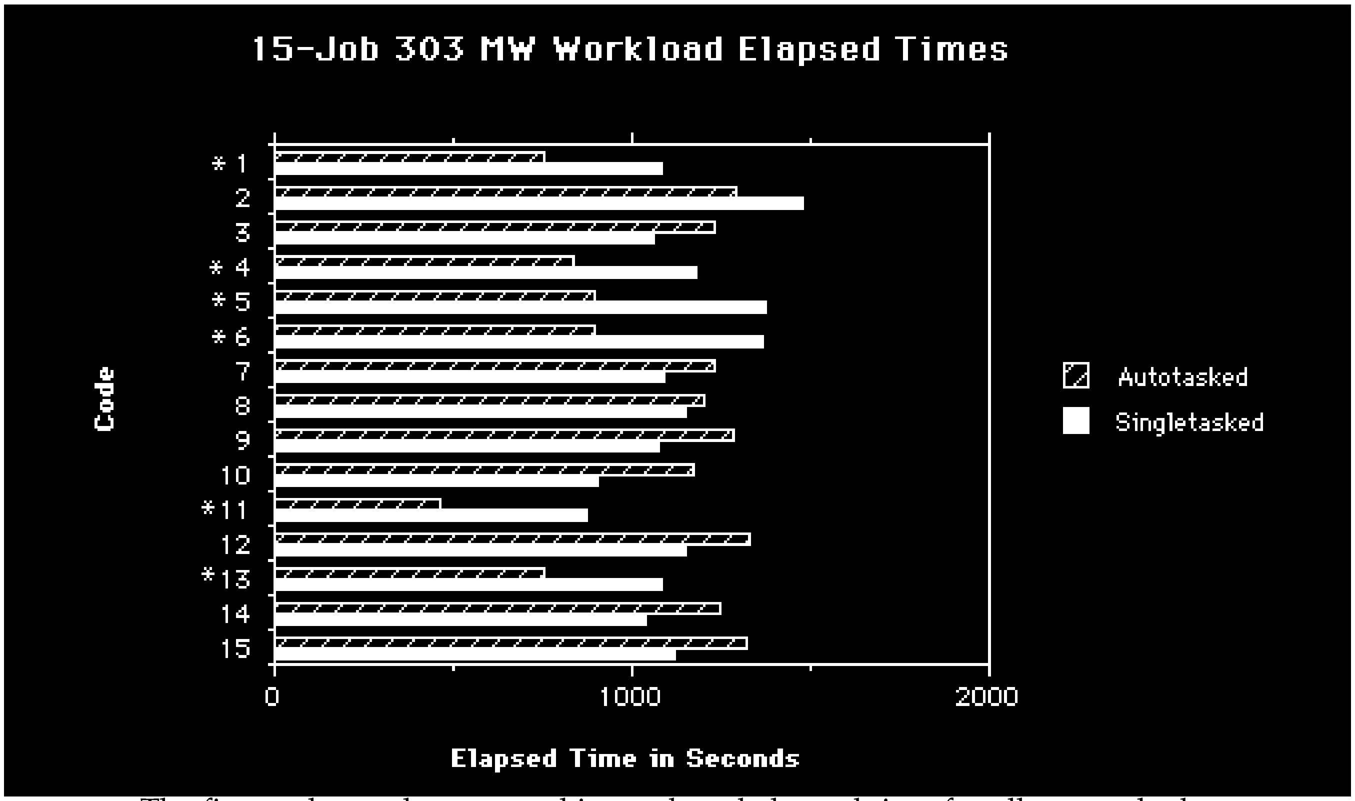
Autotasked Workload:

Maximum Throughput (Billions of floating point operations)

=1425+(1.075/8)*1812=1425+243=1688

The calculation shows that the autotasked workload can achieve about 96% of the maximum singletasked throughput.

The following figure displays elapsed times for the 303 MW workload. The figure indicates the elapsed time for each code for execution in the singletasked and autotasked workloads. Asterisks label the autotasked codes.



The figure shows that autotasking reduced elapsed time for all autotasked codes. The high efficiency displayed by the autotasked jobs, as shown in Table

3, implied a small number of self-driven I/O interrupts. These jobs displayed elapsed time reductions in the range of 50 to 70% and also tended to finish first as described in the discussion of the 287 MW workload.

4.0 Discussion

These experiments indicate that autotasking a portion of a heterogeneous workload continues to improve workload performance.

The workload represented the NAS Y-MP Off-Prime batch workload with a small interactive content, a small oversubscription of memory, performance exceeding 100 MFLOPS per CPU, and scheduler parameters designed to promote throughput of computationally intense jobs. The codes comprising the autotasking portion displayed single CPU performances exceeding 130 MFLOPS and dedicated speedups exceeding 3.

Table 6 shows workload elapsed time and its components: CPU, system and idle for the workloads described in Tables 2, 4, and 5. The CPU times quoted in the table include semaphore wait time. The data show that autotasking portions of the larger memory workloads provide increases in workload throughput relative to their singletasked counterparts.

Table 6. UNICOS Off-Prime Autotasking Summaries

Workload	12-job 169 MW		12-job 287 MW		15-job 303 MW	
Scheduler	12-job Off-Prime		12-job Off-Prime		15-job Off-Prime	
Workload	Single	Auto	Single	Auto	Single	Auto
Elapsed Time	1219	1173	1471	1212	1481	1326
CPU Time	7319	7891	6988	8157	8794	9598
System Time	96	177	88	163	404	437
Idle Time	2337	1316	4639	1376	2650	572
System GFLOPS	0.903	0.939	0.749	0.903	0.962	1.075

The throughput of the 287 MW singletasked workload is about 20% less than that of the 169 MW workload, due primarily to its increased idle time. The different order of job completion for these two singletasked workloads determines the length of time CPUs idle for lack of memory and lack of work. While the UNIX scripts preserve the initial execution order of the programs, a request to UNICOS by a program for more memory will cause the program to be swapped out of memory when insufficient memory is available. The program may not be able to return to memory until a later time and such a delayed return will change the order of execution and alter the workload performance. While the execution order for two workloads with the same memory oversubscription and the same scheduler parameters will be the same, increasing the oversubscription of a workload (e.g., 287 MW versus 169 MW) will alter the execution order because a program in the 169 MW workload which could obtain sufficient memory from UNICOS without a swap may be placed on swap when requesting more memory in the 287 MW workload. Such a request by the 105 MW C02 in the 287 MW workload forced UNICOS to place C02 on swap until sufficient memory was available and such availability occurred late in workload execution; consequently, most of the C02 execution occurred when all of the other jobs had completed.

Autotasking the 287 MW workload produced a 20% throughput increase in because the large 105 MW job was able to fit into memory much earlier in the execution of the workload.

The 303 MW singletasked workload displayed increased performance for the singletasked workload since its larger number of jobs were better able to utilize the idle time. Autotasking this workload produced a 12% throughput increase.

5.0 Conclusions

A series of model workload executions has indicated that UNICOS continues to allow autotasking codes to improve workload performance. Since these workloads performed at 100 MFLOPS/CPU prior to autotasking, this increase in performance is significant. Workload performance improved because autotasked jobs were able to transform idle cycles into cycles performing useful work. Since the workloads performed a fixed number of floating point operations, CPUs idled after the available work was exhausted and such idling would not occur in a real workload. Autotasking-generated throughput increases in a real workload should be smaller than those measured in this report. These results do show that, under the current Cray implementation, autotasking continues to be an efficient method for utilizing idle workload cycles.

NAS is currently encouraging the autotasking of large memory jobs in its workload and an important factor in user participation is the distribution of the system resources in an equitable fashion. Use of charging algorithm based on a minimum number of CPUs discourages the submittal of singletasked large memory jobs to the batch queue intended for parallel jobs. However, the system load may prevent even an efficient user application from obtaining the minimum number of CPUs, and in this case, the algorithm will penalize the user employing autotasking. Administrators should establish conditions which will ensure that the large memory multitasked jobs will obtain the minimum number of CPUS and benefit from executing their codes on multiple CPUs.

6.0 References

R. Bergeron (1990) "Performance Analysis of the NAS Y-MP Workload," NAS RND Technical Report RND 90-009.